REMARKS

Claims 1-13, 16, 22-34, 37-50 and 53 are pending in the present application. Claims 14-15, 17-21, 35-36 and 51-52 were canceled, and Claims 1-2, 9-10, 22, 30-31, 38 and 50 were amended. Reconsideration of the claims is respectfully requested.

Amendments were made to the specification to correct errors and to clarify the specification. More particularly, the specification has been amended to show the application numbers of two related patent applications. No new matter has been added by any of the amendments to the specification.

I. 35 U.S.C. § 102, Anticipation

The Examiner has rejected Claims 1-17 and 22-53 under 35 U.S.C. § 102 as being anticipated by U.S. Patent No. 6,185,574, to Howard et al. This rejection is respectfully traversed.

II. 35 U.S.C. § 103, Obviousness

The Examiner has rejected Claims 18-21 under 35 U.S.C. § 103 as being unpatentable over Howard et al. This rejection is respectfully traversed.

III. Response to Rejection of Independent Claims

In making their invention, Applicants sought to improve tracking of data and configuration files associated with application programs. Applicants recognized that this objective could usefully be achieved by establishing and maintaining a close relationship between a program and a file that the program is disposed to access. In accordance therewith, Applicants teach the following, at p. 10, lines 6-15 of their application:

> When any executing program (application, service, etc.) makes a request to open, close, delete, rename, or move a file, the request is detected, and the name of the requesting program is identified. The being operated on and the name of the program accessing the file is used to automatically create a relationship between the two. This relationship, file and program is captured and represented in a relational meta data format. Additional meta data about the file creation can also be captured, such as the location of the file, time, date, or identity of the user. This relational meta data can be stored in another data file in the file system or saved in a

> database (i.e. registry, database, directory, etc.) The database can be protected and hidden from users to prevent the deletion or corruption of the data. (Emphasis added.)

Thus, in Applicants' invention, each time an executing program requests access to a particular file, for any reason, the accessing event is captured and represented in a meta data format. This representation closely links the particular program and the particular file, and is saved in a database. Then, at a subsequent time, the database can be used to find the file names and location of all data and configuration files associated with the particular program. This is expressly taught in the application, such as at page 13, lines 8-10, as follows:

> Because this information is in database **301**, application **304** can query database **310** through calls to device driver **308** to find the file names and location of all of the data and configuration files associated with the application **304**.

To emphasize the above essential features, Applicants' Claim 1 now reads as follows:

> A method in a data processing system for tracking relationships between programs and data, the method comprising:
>
> receiving a file access request from a particular program, wherein the file access request is for a particular file and is received at an operating system level and wherein the particular file is defined by a specified file name;
>
> creating an association between the specified file name and the particular program requesting the file access, in response to receiving the file access request; and
>
> responsive to creating the association, storing the association between the specified file name and the particular program, wherein the association is used for subsequent accesses to the file and is stored in relation to other stored associations, said other associations respectively corresponding to each of the other such that a stored association is stored for each file names for which file access is requested by the particular program; and
>
> saving all of the stored associations for at least the life of the particular program.

Support for the saving step of Claim 1 is provided, for example, at page 13, lines 8-10 set forth above. In the Office Action, the Examiner stated the following in regard to Claim 1:

> 9. As to Claims 1,17,22,38, Howard teaches a system which including 'data processing system for tracking relationships between programs and data [col 8, line 41-49], Howard specifically teaches establishing relation between virtual directory system and application programs as detailed in col 8, line 41-59
>
> 'receiving a file access request from a program, wherein the file access request is for a file and is received at an operating system level and wherein the file is defined by a file name' [col 4, line 18-23, line 55-57, col 7, line 38-53], Howard specifically teaches relationship between hierarchical directory and operating system, further file attribute information especially file attribute information including file name, type of data for example text file, executable file, wordperfect file and like that corresponds to defining specific file type and name;
>
> 'creating an association between the file and the program requesting the file access in response to receiving the file access request' [col 7, line 61-67, col 8, line 1-5, line 13-15], Howard specifically suggests file information and relationships are stored in a database, further this database is configurable so that the directory can be manipulated into new configuration and can make files available at an operating system level that corresponds to creating an association between file and the program, wherein the association is used for subsequent accesses to the file such that a stored association is stored for each file name for which file access is requested by the program' [col 11, line 10-26, fig 1], Howard specifically teaches virtual directory system having data files and association between files and operating system for accessing the specific file that are stored in virtual directory system as detailed in fig 1, col 11, line 10-25.

Office Action dated February 23, 2005, pp.3-4.

Shown below are excerpts from the Howard et al. reference at col. 4, lines 18-23, col. 4 lines 55-57, col . 7, lines 27-67, col. 8, lines 1-15, col. 8, lines 41-61 and col. 11, lines 10-26. These include all the citations to the Howard specification that are set forth in the above statement of Examiner, in regard to Claim 1.

being run. The Interceptor modules then can relay the changes to the virtual directory system for organization and permanent storage. Using this method the native file system's hierarchical directory can be maintained at the same time that a parallel virtual directory system is built and updated.

When a user accesses a file via the virtual directory system the virtual directory system can retrieve the file from the appropriate storage device via the Interceptor modules.

stored on the C: hard drive). Also, a "virtual directory" is a directory of file information which can be presented to a computer operator, operating system, application program or some other aspect of a computer system as a directory that is representative of a physical file storage device(s); however, this virtual file directory is merely an apparent or "virtual" directory since it can merely store file attribute information and is not actually affiliated with an actual physical storage device, as one would associate the affiliation between one's "C: hard drive" on an IBM personal computer and the directory listing for that hard drive. However, it can be representative of a physical file storage device in that the operating system would communicate with it in the same way that the operating would communicate with a directory for an actual file storage device (e.g. in issuing open/read/write delete requests to a file system driver for a storage device.) Finally, "file attribute information" is intended to mean properties of a file in defined categories, such as, for example, file name, type of data represented by the file (e.g., text file, executable file, Wordperfect 6.1 file, etc.) file size, date and time of file creation or modification, date and time the file was accessed, programs which caused the file to be accessed, physical file storage device where file data for the file is stored, parent directory on the physical file storage device in which the file is stored, child or subdirectory for the file, user defined descriptions for the file, etc.

The virtual directory system can serve many functions and appear in different embodiments. Perhaps one of its most useful functions is the ability to store file attribute information (16) about files stored on a native file directory system (8) and present that file information as if the files were stored in a distinct directory separate from the native file directory system.

The virtual directory system is significant in that it can provide in a single database a configurable database of file information which act as links to the actual file data. No longer is a user constrained by the physical limits of a storage device as to how many files can be listed in a single directory, as is the case with a single hard drive. Also, this database is configurable so that the directory can be manipu-

lated into a new configuration and can make files available at an operating system level. This is a very significant feature, because it allows the newly configured directory to be made available at the operating system level to application level programs. In this fashion, a user could configure the virtual global directory with all of the user's word processing data files. Then, when the user wanted to find an old word processing data file, only the virtual directory would need to be accessed. This would save the user from having to search through all of the various directories on the computer system until the desired word processing document was located.

Similarly, the virtual directory system allows a user to store all of the files on the computer system in a single directory, if desired. This can be accomplished because the

Furthermore, because the virtual directory system can be stored on a configurable database, the directory itself can be subdivided into additional virtual directories. Hence, a user can group all of the word processing applications into a single directory, all of the letters to relatives (which also are stored in the word processing directory) in a second directory, and all of the letters to Aunt Margaret (which are also stored in each of the first two directories) in a third directory. Then, each of these directories can be utilized by application programs as if they were directories of actual physical file storage devices, such as a local hard drive. The application programs which utilize the file information do not need to be able to configure the database of the virtual directory, rather the directory can be configured at the operating system level and therefore it can appear as a physical storage device when called by the application level programs that need files represented by the virtual directory system. (It should be understood, however, that the virtual file directory system can be configured by an application level program. Therefore, the configuration can be accomplished quite easily when a new configuration is desired.)

An additional connection (201) is shown in a dashed line between the virtual directory system and the I/O system of the operating system. In FIG. 1, the I/O system is shown in part as the Installable File System Manager. This connection or coupling shows that the virtual directory system might transfer data and commands back up to the application level to input and output information through the I/O system. As a simple example, the virtual directory system when accessed for a file shown on its virtual directory listing can locate the actual physical storage location for the requested file and initiate its own request back through the installable file system. In this manner, the virtual directory system can request a file from the C: drive, for example, then present it to a word processing program as actually having come from the virtual file directory itself. In this fashion, the virtual directory system can emulate or act as an actual physical storage device

The above excerpts from column 4 of Howard appear to disclose that a virtual directory system can retrieve files from a storage device. The excerpt from col. 7, at lines 37-52, teaches that a virtual directory can be presented to an operating system, an application program or the like, and can be representative of a physical file storage device. The virtual directory can store file attribute information, which is defined to be properties of a file in defined categories. At col. 7, line 61-col 8, line 15, Howard teaches that the virtual directory can be made available at the operating system level to application level programs. Also, the virtual directory could be configured with all of a user's word processing files, so that only the virtual directory would need to be accessed to find old files. The excerpt at col. 8, lines 41-61 discloses that a virtual directory can be subdivided into additional directories, which can be used by application programs as if they were directories of actual physical storage devices. The excerpt at col. 11, lines 10-26 refers to an Installable File System Manager of FIG. 1, and indicates that the virtual directory system can locate the actual physical storage location for a requested file and can initiate its own request.

A prior art reference anticipates a claimed invention under 35 U.S.C. § 102 only if every element of the claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F. 2d 831, 832, 15 U.S.P.Q. 2d 1566, 1567 (Fed Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F. 3d 1579, 1582, 21 U.S.P.Q. 2d 1031, 1034 (Fed Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. Kalman v. Kimberly-Clark Corp., 713 F. 2d 760, 218 U.S.P.Q. 781 (Fed Cir. 1983).

Applicants respectfully submit that Howard does not teach every element of Claim 1, arranged as they are therein. Specifically, Howard does not teach the creating, storing and saving steps now recited in Claim 1, either in the over-all combination of Claim 1, or in combination with one another. These steps recite the following features or limitations.

(1) An association is created between a specified file name and a particular program in response to receiving a file access request.

(2) In response to creation of the association, the association between the specified file name and the particular program is stored, in relation to other stored associations that likewise pertain to the particular file.
(3) All of the stored associations are saved, for at least the life of the particular program.

The Howard reference, as discussed above, teaches that a virtual directory system can generally provide a database of file information, or can locate the physical storage location of a requested file. However, Howard does not show, either in the excerpts discussed above or elsewhere, the above features (1)-(3), carried out in combination with one another. More particularly, Howard does not show that in response to receiving a file access request, an association between a particular program and specific file name is created, is stored in relation to other such associations, and is saved for the life of the particular program, or other specified period of time. Even if Howard taught some of the elements of Claim 1, but not all elements in the claimed relationship, it would still not be anticipatory of Claim 1.

Independent Claims 22 and 38 recite subject matter similar to subject matter of Claim 1, and are each considered to patentably distinguish over the art for the reasons given in support for Claim 1.

IV. Response to Rejection of Remaining Claims

Claims 2-13 and 16 respectively depend from independent Claim 1, and are each considered to patentably distinguish over the prior art for the same reasons given in support thereof.

Claim 2 is considered to further distinguish over the prior art in reciting that the method includes accessing a database containing each of the stored associations to find the file names and locations of all of the data and configuration files associated with the particular program. Applicants consider that Howard does not teach or suggest this feature.

Claims 23-34 and 37 respectively depend from independent Claim 22, and are each considered to patentably distinguish over the prior art for the same reasons given in support thereof.

Claims 29-50 and 53 respectively depend from independent Claim 38, and are each considered to patentably distinguish over the prior art for the same reasons given in support thereof.

Conclusion

It is respectfully urged that the subject application is patentable over Howard et al., and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: April 15, 2005

Respectfully submitted,

James O. Skarsten

James O. Skarsten
Reg. No. 28,346
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants